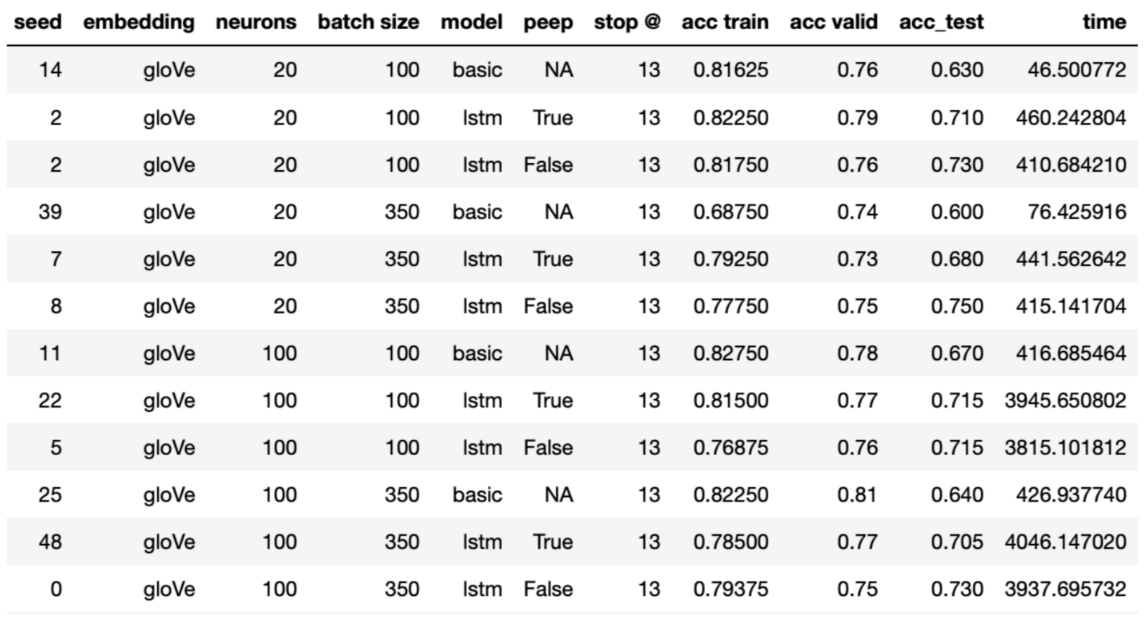
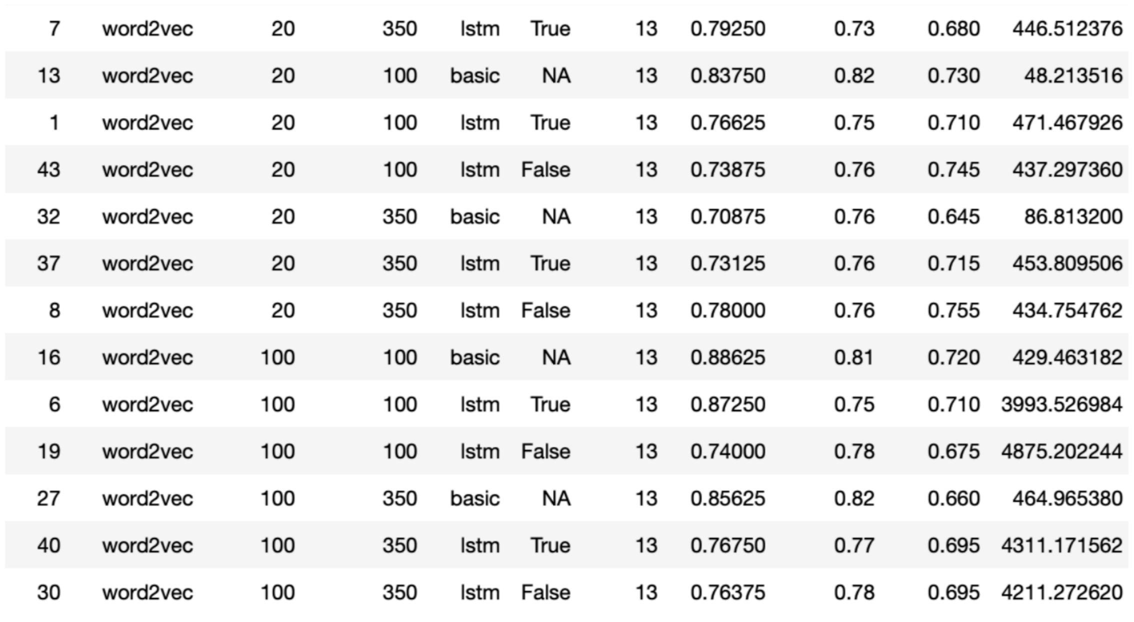
Summary: This summarizes a benchmark test comparing accuracy from recurrent neural network language (RNN) models configured **24 different** ways to predict a binary context = negative or positive movie reviews for sequences of 40 target words from 1000 reviews.



seed	embedding	neurons	batch size	model	peep	stop @	acc train	acc valid	acc_test	time
14	gloVe	20	100	basic	NA	13	0.81625	0.76	0.630	46.500772
2	gloVe	20	100	lstm	True	13	0.82250	0.79	0.710	460.242804
2	gloVe	20	100	lstm	False	13	0.81750	0.76	0.730	410.684210
39	gloVe	20	350	basic	NA	13	0.68750	0.74	0.600	76.425916
7	gloVe	20	350	lstm	True	13	0.79250	0.73	0.680	441.562642
8	gloVe	20	350	lstm	False	13	0.77750	0.75	0.750	415.141704
11	gloVe	100	100	basic	NA	13	0.82750	0.78	0.670	416.685464
22	gloVe	100	100	lstm	True	13	0.81500	0.77	0.715	3945.650802
5	gloVe	100	100	lstm	False	13	0.76875	0.76	0.715	3815.101812
25	gloVe	100	350	basic	NA	13	0.82250	0.81	0.640	426.937740
48	gloVe	100	350	lstm	True	13	0.78500	0.77	0.705	4046.147020
0	gloVe	100	350	lstm	False	13	0.79375	0.75	0.730	3937.695732



7	word2vec	20	350	lstm	True	13	0.79250	0.73	0.680	446.512376
13	word2vec	20	100	basic	NA	13	0.83750	0.82	0.730	48.213516
1	word2vec	20	100	lstm	True	13	0.76625	0.75	0.710	471.467926
43	word2vec	20	100	lstm	False	13	0.73875	0.76	0.745	437.297360
32	word2vec	20	350	basic	NA	13	0.70875	0.76	0.645	86.813200
37	word2vec	20	350	lstm	True	13	0.73125	0.76	0.715	453.809506
8	word2vec	20	350	lstm	False	13	0.78000	0.76	0.755	434.754762
16	word2vec	100	100	basic	NA	13	0.88625	0.81	0.720	429.463182
6	word2vec	100	100	lstm	True	13	0.87250	0.75	0.710	3993.526984
19	word2vec	100	100	lstm	False	13	0.74000	0.78	0.675	4875.202244
27	word2vec	100	350	basic	NA	13	0.85625	0.82	0.660	464.965380
40	word2vec	100	350	lstm	True	13	0.76750	0.77	0.695	4311.171562
30	word2vec	100	350	lstm	False	13	0.76375	0.78	0.695	4211.272620

Research design: 20 first and last words from 500 negative and 500 positive movie reviews are parsed, then encoded with (1) two 300-dimension **embeddings**: a 1 million word2vec Google News and a 400,000 word gloVe Wikipedia vocabulary are trimmed to 300,000 words. Sample reviews are split randomly 800/200 train/test and then 700/100 train/validate. (2) **Batches** of 100 or 350 instances randomly indexed from train are fed to (3) 3 different **models**: (a) a single layer basic RNN and 2 long short-term memory (LSTM) cells (b) with and (c) without peepholes. The 350-row batches provide more training data but cycle only twice (700/350) per epoch while 100-instance batches cycle 10 times. Atop basic-cell tanh-activation, LSTM's 3 layers' of logisitic gates control long-term state forgetting and retention and blending with its basic cell output. Peepholes allow long-term state to leak into the input and short-term state prior to activation. (4) Basic and LTSM layers are equipped with either 20 or 100 **neurons** and (5) initialized with 50 different **random seeds**. Adam then minimizes mean x-entropy loss and the best seed is kept. (6) **Early stopping** is simulated by saving the lowest loss of 50 cross validations, but all 50 epoch accuracy figures are plotted in learning curves at the bottom of this document. **Table 1** above presents test combinations' results.

Data exploration: (1) After reduction to 300,000 words there are 127,529 words in word2vec set that are not in in the gloVe set and 170,400 words in gloVe not in word2vec. (2) Reviews vary from 22 to 1052 words and reviews share similar dimensions: 107,031 total words in positive reviews and 106,047 in negative reviews, 214 and 212 words on average per positive and negative review, minima of 26 and 22 words per review and maxima of 1052 and 972 words per review. (3) only 10,111 of the 107,031 positive and 10,072 of the 106,047 negative movie review words are found in the word2vec embeddings; 5643 of those overlap: thus, about half the 10k words identified by the word2vec embeddings would confuse the model. gloVe has similar figures: 10,335 positive, 10283 negative review words are found in gloVe and 5659 overlap. (4) the shortest reviews must double count words in their sequences: a 20 words selected from the front and back of a 22 word review leaves all but 2 double counted in a 40 word sequence. On the positive side, shorter reviews have a larger proportion of their words (and sentiment) reflected in their instances.

Python code: (1) Furnished code downloads gloVe and word2vec embeddings, presorted for word frequency. (2) Furnished code transforms space-delimited lines of gloVe word / vector into a dictionary of pairs, key = word / value = index, and a corresponding indexable array of embeddings. The word2vec embeddings arrive in binary dictionary form; furnished code, amended, generates the same data structures. (3) Dictionaries and arrays assign embeddings a zero vector for unrecognized words, via defaultdict. (4) An amended standard sentence tests embeddings functionality and discovers word2vec does not encode the article ‘a’. (5) Both embeddings are trimmed for their first = most common 300,000 words, original files are deleted to conserve memory, and trimmed embeddings retested against a standard sentence. (6) Stopwords are not removed from movie reviews but return characters are, words are made lowercase and tokenized with TreebankWordTokenizer. The fashion with which this function parses "They'll" into "They" and "'ll" may explain why so few embeddings and review words match (see EDA point 3). (7) The first and last 20 words from 1000 reviews are retrieved via the itertools’ chain function and encoded into the 2 different embeddings. The mappings of words to embeddings is confirmed again. Files are saved as numpy arrays, dimensioned 1000 reviews x 40 words x 300 embeddings, and renamed ‘embeddings_array’ (for gloVe Wikipedia) and ‘embeddings_array_g’ (for word2vec GoogleNews). (8) 1000 labels are assigned in the same order: 500 negative (zeros) followed by 500 positive (ones). (9) sklearn randomly splits 1000 sequences and labels 80/20. The 800 train are sequentially split into sets of 700 train / 100 validate; sklearn's initial random split may suffice for randomizing these development sets. The set of the original 800 train_expanded is retained to collect accuracy measures across the original train set. (10) a directory is set for saving sessions to accommodate proxy early stopping. (11) the embeddings 2nd and 3rd dimensions, 40 words and 300 embeddings, are assigned to n_steps and n_inputs. Fixed are n_outputs = binary = 2, learning rate = 0.001. Placeholders are set. (12) **neurons** is a **parameter** = 20 or 100. (13) 3 **models** are constructed with logit outputs and included as a **parameter**: 'basic', 'lstm' with peepholes, 'lstm' without peepholes. (13) A shufflebatch routine is defined; **batch size** varies as a **parameter** 100 or 350. (14) epochs are fixed at 50, but as an epoch's x-entropy improves the minimum, a session is saved and then retrieved when 50 epochs are finished. (14) all epochs train and test accuracy are retained for learning curve analysis. (15) These 24 configurations run 50 times, once for each **seed** **parameter** 0 to 49, and the best seed's results are kept: accuracy measures for train, validate, test,and learning curves. The code is tested in modular form, but then since embeddings is a parameter, and elapsed time calculated, the above code from step 9 onwards is wrapped in a defined function contained in for loops from outside to inside as follows:

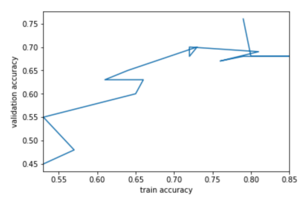
Embeddings: gloVe and word2vec → neurons: 20 and 100 → Batch size: 100 and 350 → Rnn_model: 'basic', 'lstm', 'lstm with peepholes' → 50 seeds: zero to 49

Results: Results shown in **table** 1 above demonstrate (1) the LTSM model with only 20 neurons and peepholes off, after 15 of 50 epochs, employing 350-instance batch sizes recorded the highest test **accuracy** = 0.75 with gloVe and 0.755 with word2vec. (2) **Speed**: while LSTM models are 10x as expensive as basic models, models with 5 times as many neurons (20 → 100) are 10x slower again: LSTM with 100 neurons is 100x slower than basic with 20 neurons. Batch size has little impact on speed. (3) **<u>Generalization</u>**: neuron heavy models have higher **average** train and validation and similar test accuracy and generalization in the form of declines from train to validation to test: from 0.808 to 0.779 and 0.694 for 100 neurons versus 0.773 to 0.761 and 0.700 for 20 neurons. LSTM underperforms basic on validation 0.762 vs .788 but outperforms in test 0.714 versus 0.662. (4) Smaller **<u>batch sizes</u>** generally produce better average accuracy: 0.705 for 100 samples per batch and 0.689 for 350. (5) **<u>Vocabularies</u>**: word2vec average test accuracy 0.704 outperforms gloVe 0.689 and this margin is greater for 20 neurons cells: 0.716 versus 0.683 on average. There's virtually no performance difference between embeddings for models with 100 neurons (6) **<u>Early stopping</u>**: nearly all models were near 100% train accuracy after 50 epochs; learning curves show clear overfitting after early-stopping. No pattern can be discerned from earlier versus later stopping models, but all stopped after no more than 15 iterations. Considering that train accuracy is bordering 100% after 50 iterations, extending iterations past 15 is not advised, but some method alteration seems in order to cause the model to iterate longer before finding a minimum loss. Other forms of initialization, activation, regularization should be investigated.

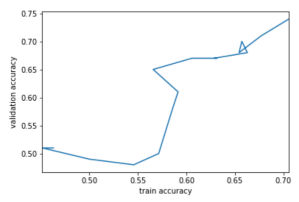
Management recommendations. Parameter selection will depend upon many factors. (1) Select **LSTM** over basic. Where basic performs well in validation, it slips into 60s accuracy in test, which would surprise. (2) If neuron count bubbles to the top of selection criteria, choose **fewer neurons when selecting word2vec embeddings**. (3) Larger batch sizes generally fare worse in accuracy measures versus smaller batch sizes. (4) Generally, choose **word2vec**. Atop a an accuracy margin, Word2vec has the additional advantage of scaling vocabulary to 1 million words. (4) Run **random sampling of seed values** when solving for a predictive model: there are no patterns discernable for seed values, except that they are random. All the above recommendations will speed the model except LSTM.

Learning curves:

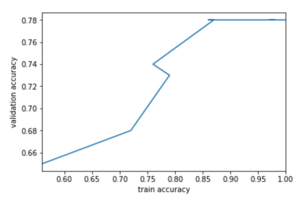
gloVe basic 20 neurons 100 batch size NA peep



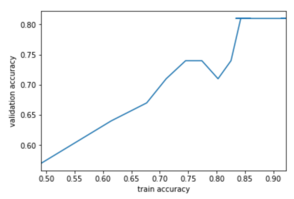
gloVe basic 20 neurons 350 batch size NA peep



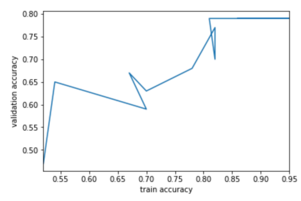
gloVe basic 100 neurons 100 batch size NA peep



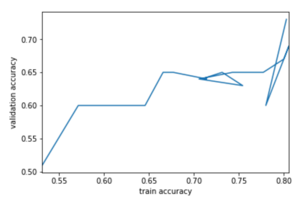
gloVe basic 100 neurons 350 batch size NA peep



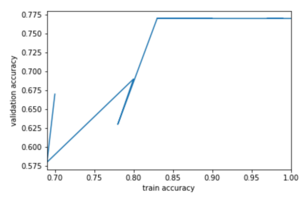
gloVe lstm 20 neurons 100 batch size True peep



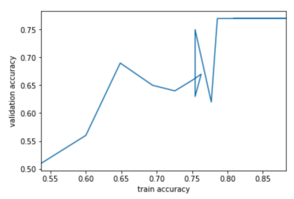
gloVe lstm 20 neurons 350 batch size True peep



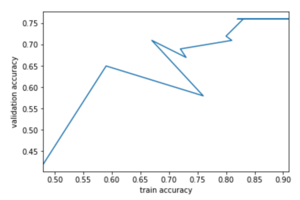
gloVe lstm 100 neurons 100 batch size True peep



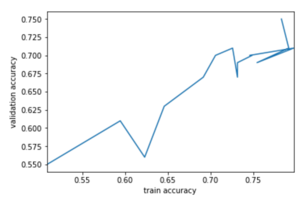
gloVe lstm 100 neurons 350 batch size True peep



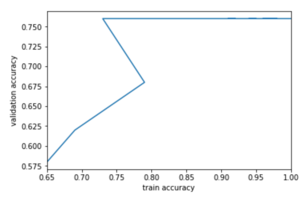
gloVe lstm 20 neurons 100 batch size False peep



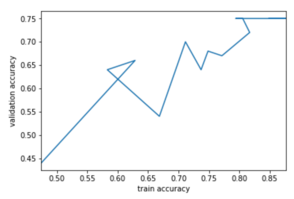
gloVe lstm 20 neurons 350 batch size False peep



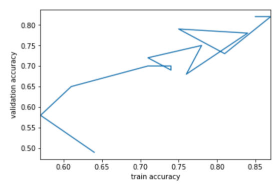
gloVe lstm 100 neurons 100 batch size False peep



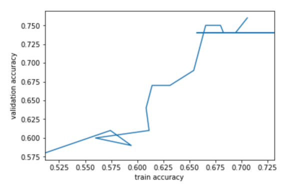
gloVe lstm 100 neurons 350 batch size False peep



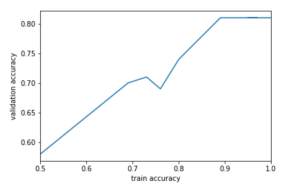
word2vec basic 20 neurons 100 batch size NA peep



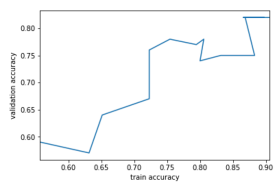
word2vec basic 20 neurons 350 batch size NA peep



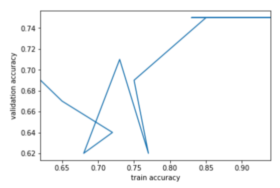
word2vec basic 100 neurons 100 batch size NA peep



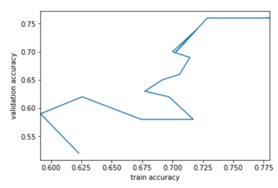
word2vec basic 100 neurons 350 batch size NA peep



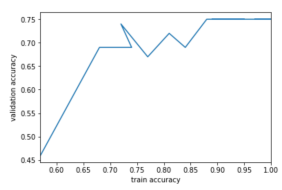
word2vec lstm 20 neurons 100 batch size True peep



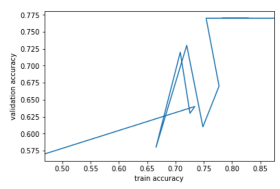
word2vec lstm 20 neurons 350 batch size True peep



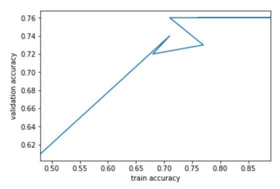
word2vec lstm 100 neurons 100 batch size True peep



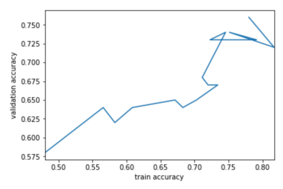
word2vec lstm 100 neurons 350 batch size True peep



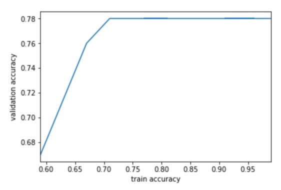
word2vec lstm 20 neurons 100 batch size False peep



word2vec lstm 20 neurons 350 batch size False peep



word2vec lstm 100 neurons 100 batch size False peep



word2vec lstm 100 neurons 350 batch size False peep